A 4-Bit Counter Using Relays

Peter Hiscocks

Department of Electrical and Computer Engineering
Ryerson Polytechnic University

phiscock@ee.ryerson.ca

November 8, 2001

Contents

1 Introduction

Every student of electrical engineering knows that logic circuits can be created by switches and relays. Combinational circuits – those that produce a certain digital output as the result of some combination of inputs – are not difficult to imagine or design. A particularly charming description of these circuits, applied to various puzzles, is given in reference [1].

However, a computing circuit require some sort of *state machine*, where the next state of the machine depends on some combination of inputs and the current state. For example, the *Turing Machine*, a mathematical abstraction of the computer, is a state machine. The common *binary counter* is another example. It's not immediately obvious how such a machine can be constructed using relays.

This paper describes a 4-bit binary counter using relays. The completed counter is shown in figure 1.

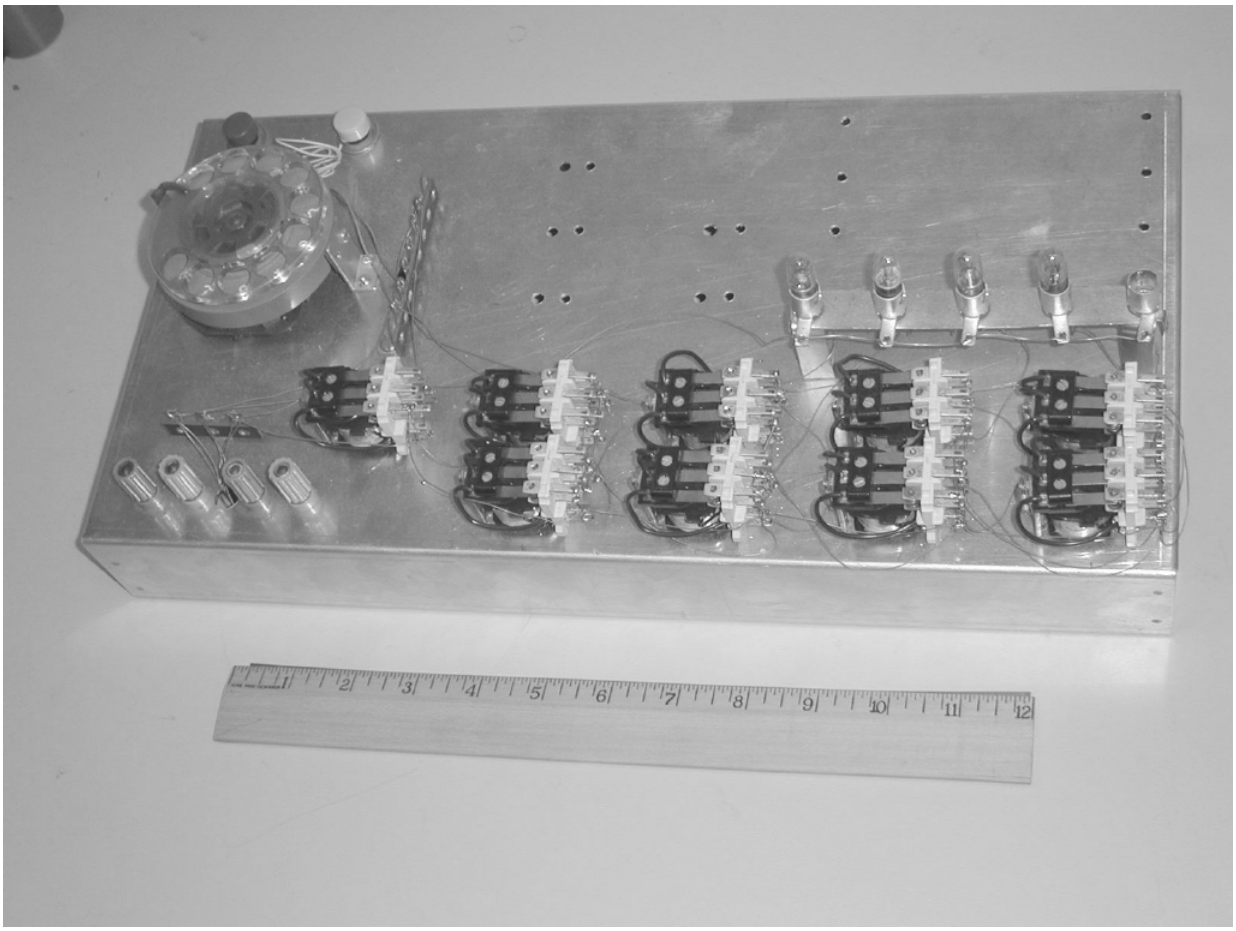


Figure 1: The Relay Counter

A drawing of the panel layout is shown in figure 2. A pushbutton resets the counter to zero. A telephone dial mechanism at the right side allows the operator to dial in various numbers, up to a total of 15. The counter relays are RY1 through RY8. RY9 is used to reset the counter. The readout of the count is shown on four lamps of the light bar LB1.

The remainder of the circuit is intended as a 1:16 decoder, but was not constructed. (It would have put the circuit beyond the current capability of the power supplies at my disposal.)

In the interest of focussing on the Main Idea, I have assumed that the reader can translate schematic diagrams

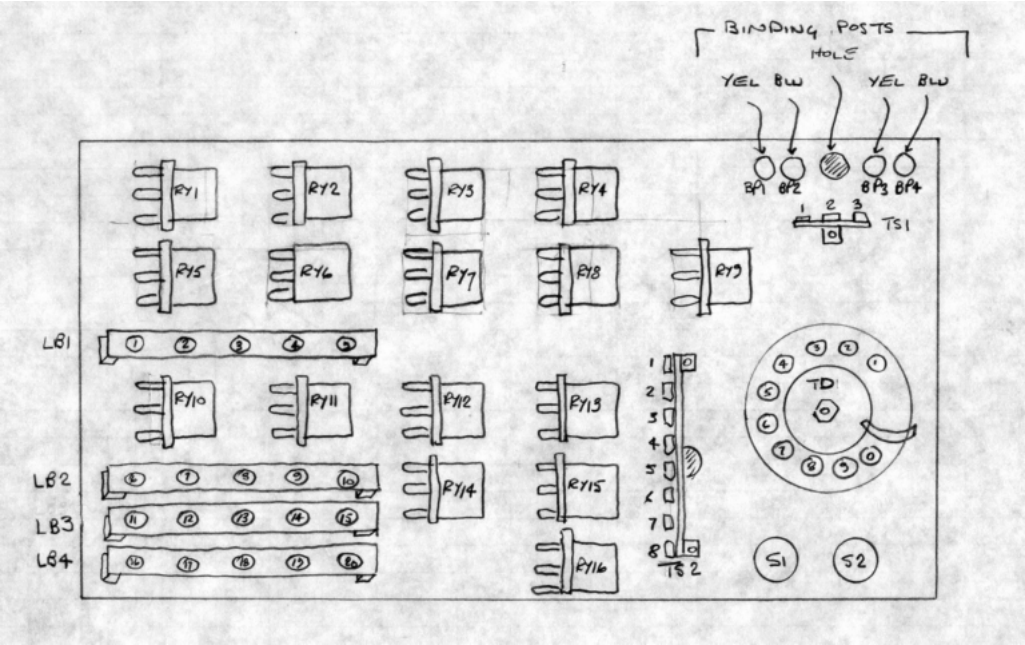


Figure 2: The Relay Counter

into wiring diagrams and is familiar with the operation of a relay.

However, since this material may be of interested to the talented and inquisitive beginner, I will supplement these notes with more detailed tutorial material if there is sufficient interest. (For example, the ideas might be useful in a science fair project on computing.) If you wish to encourage this, send me email.

Why Does The World Need This?

The relay counter is a dramatic illustration of the progress of computing technology since the era of relay computers, circa 1940 or so. It is huge by comparison with modern counters and consumes orders of magnitude more electrical current.

It is also an entertaining example of a digital circuit. Nothing is hidden. An observer can easily see the circuit in action. It's noisy and things move. This is much more interesting than an integrated circuit driving light emitting diodes.

2 The Relay

A close-up photograph of the relays used in this project is shown in figure 3. A diagram of the relay is shown in figure 4.

The relay schematic symbol is shown in figure 5.

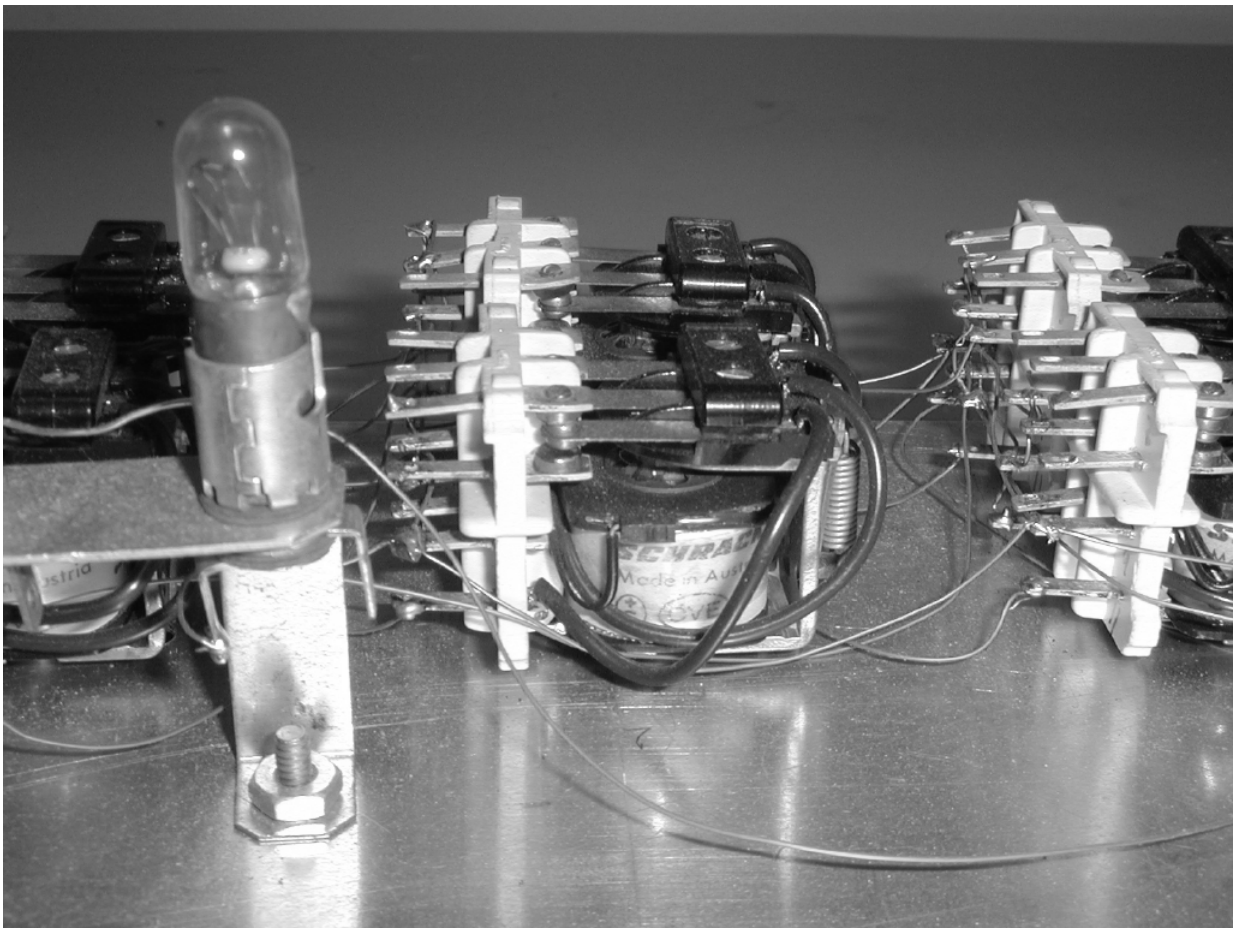


Figure 3: Relay Closeup

Each relay is 3PDT (3 switching contacts, each with two positions), with a nominally 6 volt coil. The relay terminal connections are shown in schematic form and physically on the relay, in figure 6.

These relays were used because they were readily available surplus at very low cost. Others would undoubtably do as well.

3 The Relay Flip-Flop

A *flip-flop*[1] is an electronic circuit that has two stable states. A simple relay flip-flop shown in figure 7.

When this circuit is first turned on, both relays are in the de-energized (*relaxed*) state, and the switch contacts

[1] Also known as a *bistable* circuit. This is actually a more precise name, if somewhat less colourful, because it contrasts this circuit with its brethern, the *astable* and *monostable.*

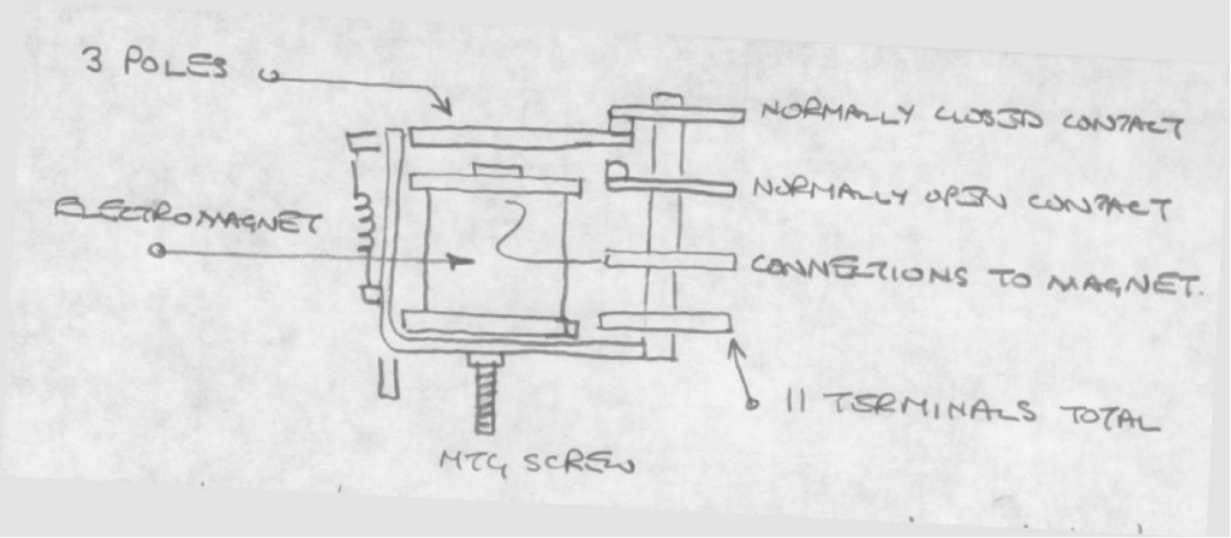


Figure 4: Relay Diagrams

are in the upper position. Consequently, both relay coils are energized, and both relays will begin to close their respective switches. However, the switches are not precisely identical and one switch will open before the other.

Let us assume that the relay B switch is the first to open. This disables power to the coil of relay A, ensuring that relay B stays activated and relay A stays de-activated. This state of affairs is one of the two stable states of the flip flop.

Now suppose you force down the contact of relay A by pressing on the switch part. This must de-energize relay B, which then supplies power to the relay A coil, ensuring that it stays de-energized. This is the other stable state of the flip flop.

You can go back and forth between the two switch sections of the relay. When you press down the de-activated switch, it will *latch* the flip-flop into the opposite state.

To indicate the state of the flip flop, you could wire two lamps, each in parallel with one of the relay coils. To switch state electrically, you could add switches in series with the power input to each of the two relays.

Toward a Toggle Flip Flop

A 4-bit binary count sequence is shown in figure 8, where A is the least significant bit.

The least-significant stage (A) toggles (changes state) with each clock pulse. For the other stages, a given stage toggles whenever all the previous stages are at logic 1.

The logic for this is often implemented using a *master-slave* flip flop. The master-slave flip-flop consists of two flip-flop latches similar to figure 7, plus some steering gates. The first latch is known as the *master* and the second the *slave*. Data is transferred into the master on one phase of the clock, and then to the slave on the other phase. (Any textbook on Digital Circuit Design will have a description of the master-slave flip flop.)

The brute force way implementation of this in relay logic might work, but it would require a huge number of parts. Each flip-flop would require two latches plus the steering logic, for at least 6 relays. A four-bit counter

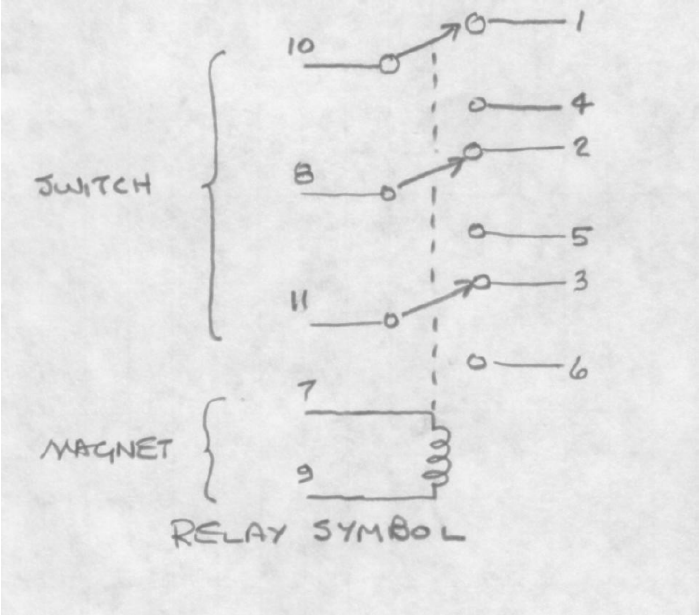


Figure 5: Relay Schematic

would require 24 relays. A simpler circuit is in order.

4 A Toggle Flip Flop

The previous flip-flop was somewhat like a wall switch: when you move it to the ON position, it stays there. That is, it remembers its setting, and it's known as a *latch* or *set-reset* flip-flop.

The *toggle* flip-flop, on the other hand, is like a pull-cord switch: one pull turns the light on, the next turns it off. The switch alternates (toggles) between states.

It turns out that the toggle flip-flop is ideal for constructing a binary counter. A toggle flip-flop, using only two relays, is shown in figure 9[2].

Basically, this circuit will toggle back and forth between two states every time the pushbutton is pressed and released. The operation is somewhat involved, so we'll work through it step by step.

1. When the power is first turned on, the circuit is as shown in figure 9. There is no path for current through either relay, so both are de-energized.

2. The pushbutton is actuated. Current flows from the supply through contact A2-A8, through the pushbutton, through contact A10-A1, and energizes the relay coil B. This causes the B relay to actuate, so the contact B10-B4 closes. This configuration is shown in figure 10.

[2]If you have trouble reading the schematics, send me a self-addressed envellope and I will send a more legible copy.

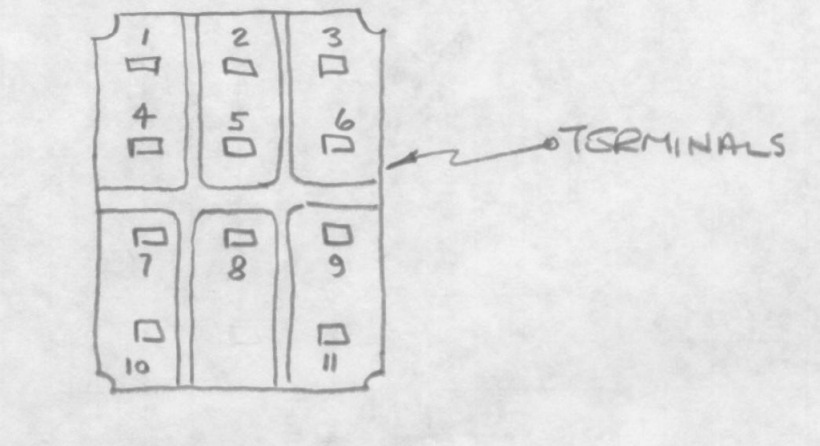


Figure 6: Relay Terminal Connections

3. Now the pushbutton is released. Because relay B is energized and switch B is closed, the opening of the switch diverts current through the coil or relay A. Current flows through the coil of relay A, contact B10-B4 and coil B.

4. The energizing of relay A causes the A switch contacts to close, as shown in figure 11.

   This is stable state zero: the flip-flop will stay in this configuration indefinitely.

Now consider what happens when the pushbutton is actuated again.

1. The current flows through the coil of relay A, contact A4-A10, the pushbutton and contact A8-A5, as shown in figure 12.

2. This bypasses the current around the coil of relay B, de-energizing it. The switch B10-B4 now opens.

3. Next, the pushbutton is released. Current stops flowing through the coil of relay A and the contacts of A4-A10 and A5-A8 open. We are back to the original configuration of figure 9. This is stable state 1. Again, the flip-flop will stay in this configuration indefinitely.

The key feature of this circuit is this: the flip flops change state through one complete cycle every time the pushbutton is actuated twice (ie, goes through two state changes).

Pushbutton State	Flip-Flop State
On	-
Off	0
On	-
Off	1

The pushbutton switch can be replaced with a relay contact to a previous stage toggle flip-flop. Then each stage will toggle at half the rate of its predecessor, and together they will count with a binary sequence.

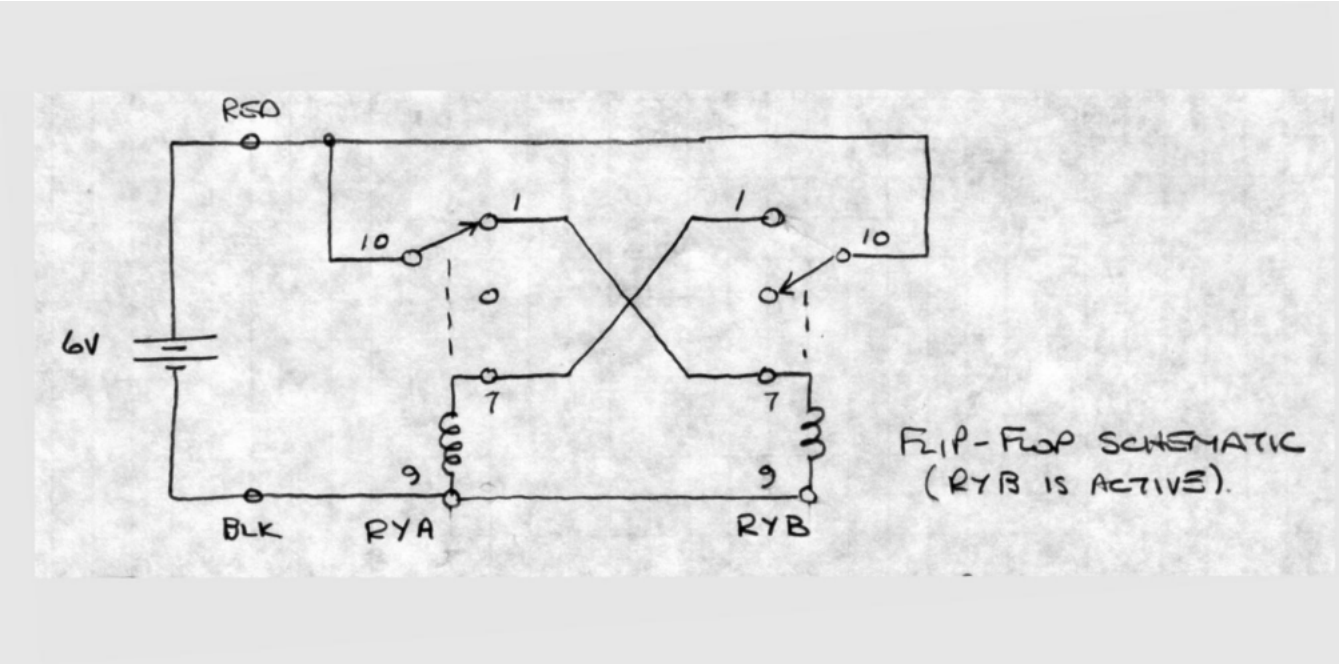


Figure 7: Simple Flip Flop

There are a couple of other interesting features of this circuit. First, notice that the flip-flops are either both energized or both de-energized: they do not alternate, as one might expect.

Second, the circuit relies on a feature of the relays: the *hold* current is much less than the *pull-in* current. It takes a certain amount of current through the relay coil to get the switch moving toward the relay core. However, once the switch is actuated, the magnetic circuit is closed, there is no gap to overcome, and so the current can decrease significantly before the switch releases. This provides the relay with a certain amount of hysterisis in its operation. The circuit takes advantage of this effect. Once relay B is actuated, it can keep B actuated even though its coil is in series with the coil of relay A.

I'd like to take credit for this clever design but must decline – it's from an old book on relay circuit design, reference now lost.

5 The Counter Circuit

The complete counter schematic is given in figure 13.

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Figure 8: Binary Count Sequence

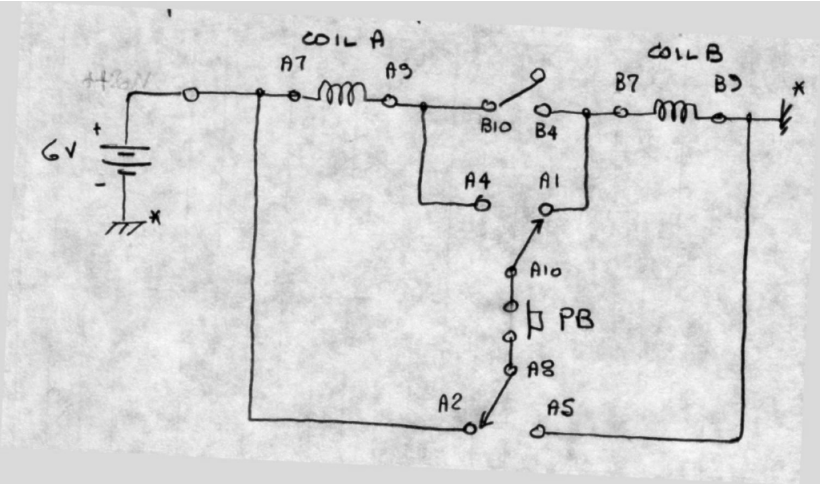


Figure 9: Toggle Flip Flop

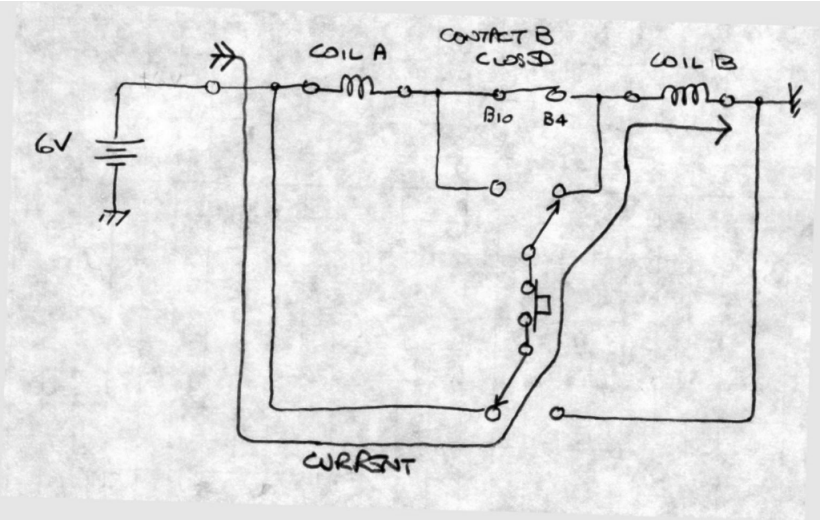


Figure 10: Toggle Sequence: Button Pressed

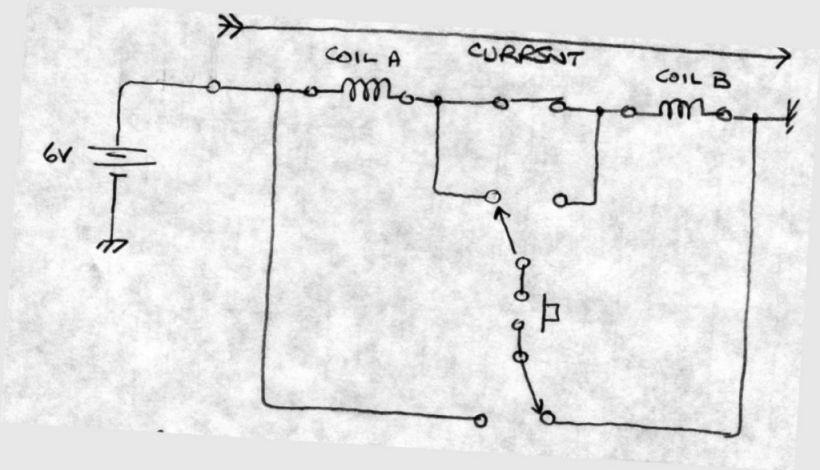


Figure 11: Toggle Sequence: Button Released

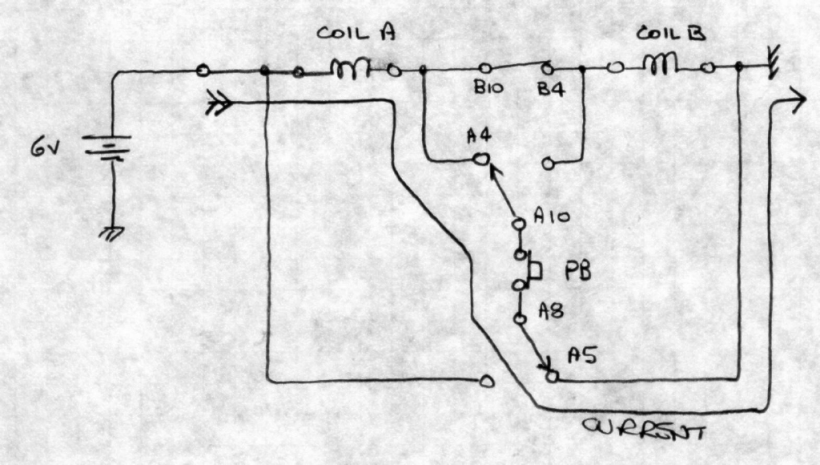


Figure 12: Toggle Sequence: Button Pressed Again

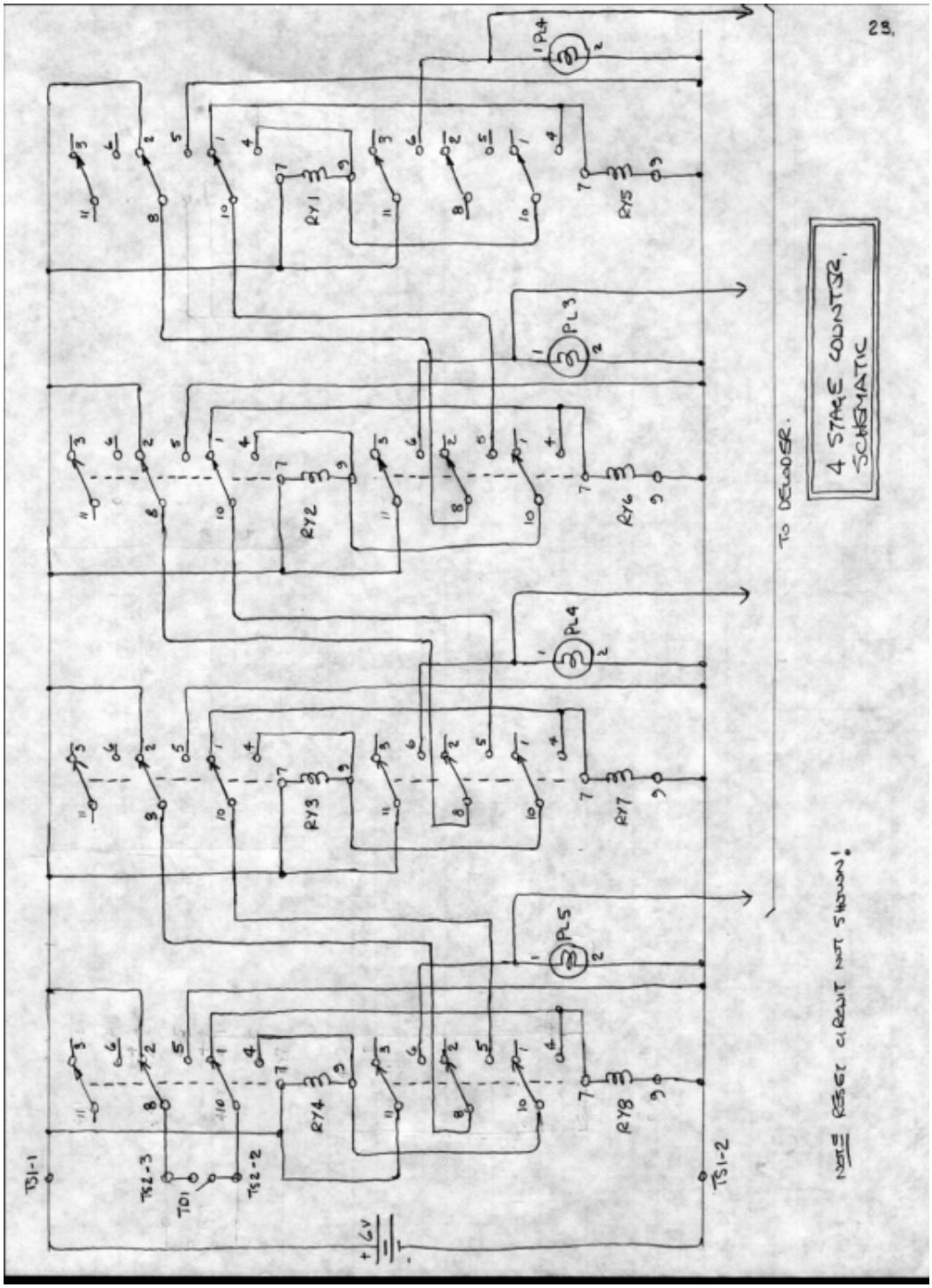


Figure 13: Counter Schematic

- The first stage of the counter is triggered by `TD1`. This could be a momentary action pushbutton. However, a telephone dial mechanism makes a more impressive input switch. Dialing a given number causes the dial mechanism to generate that number of pulses. So you can dial in one number, such as 4, and then a further number, such as 6, and the counter should show the binary value of their sum, 10.

  The rotary dial mechanism is itself a clever piece of mechanical engineering. Its speed is governed, so that it produces equally spaced pulses as it returns to the rest position.

- Each of the three subsequent stages of the counter is driven by a relay switch contact in the previous stage. For example, the input switch for the second stage is RY8 contacts 8 and 5.

- Each counter stage drives an indicator lamp through a spare relay contact, so the count can be read out as a binary number between 0 (`0000) and 15 (1111).`

6 Resetting the Counter

The counter may be reset by removing power. In this circuit, the power is supplied via the contacts of yet a *reset* relay, as shown in figure 14. When the reset button is pushed, this energizes the reset relay, which removes power from the rest of the circuit.

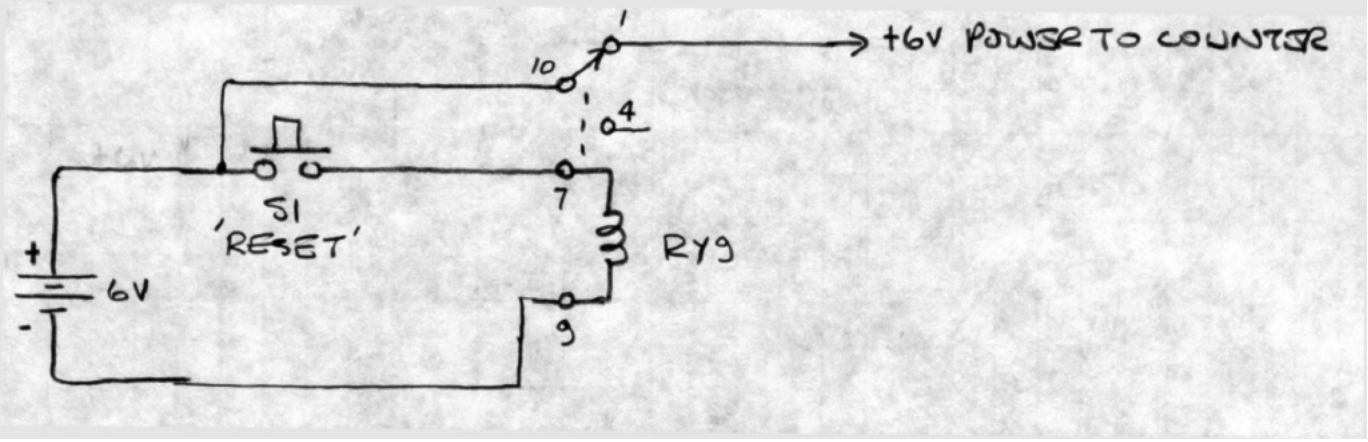


Figure 14: Reset Circuit

7 Binary to Decimal Decoder

The binary count may be decoded into a 1:16 display by a further *decoder tree* circuit. I didn't get around to wiring this up, but the schematic is shown in figure 15. It requires a further 7 relays and 15 indicator lamps, assuming that the zeroth state does not have an indicator.

If the relays had enough poles only 4 relays would be required. However, since I was working with relays that had three poles, I needed to use three relays driven simultaneously to make the fourth-level (rightmost) switch,

which requires 8 poles. I also needed to use two relays driven simultaneously to make the third-level switch, which requires 4 poles.

This circuit has other applications. It could be reversed so that it has 16 inputs and one output. The inputs are be connected to some arbitrary combination of logical one and logical zero levels. Then the circuit becomes a function generator or lookup table (*lut*) device. The inputs of the lut are the four logic signals A,B,C and D – the single output is a function of those four inputs.

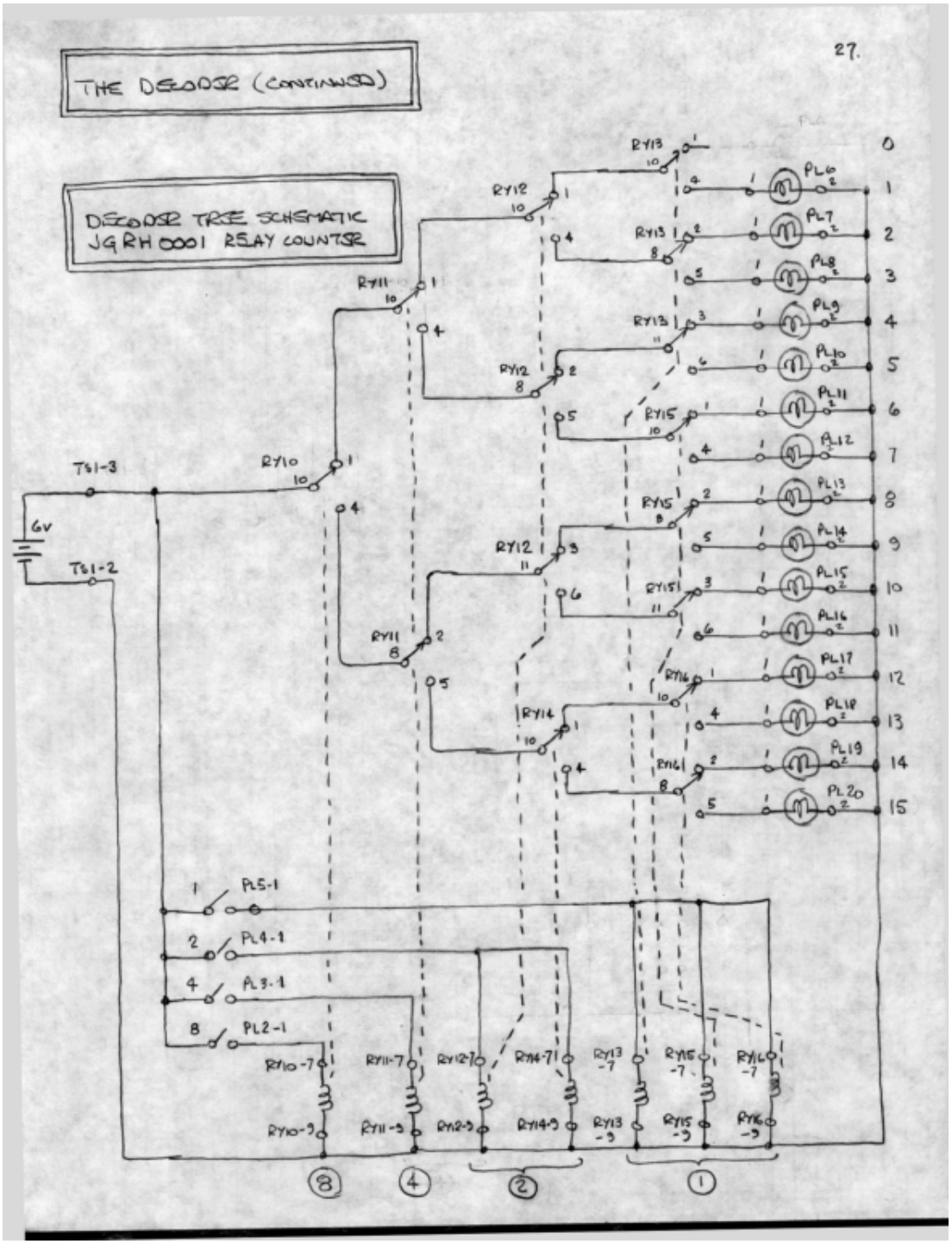


Figure 15: Decoder Schematic

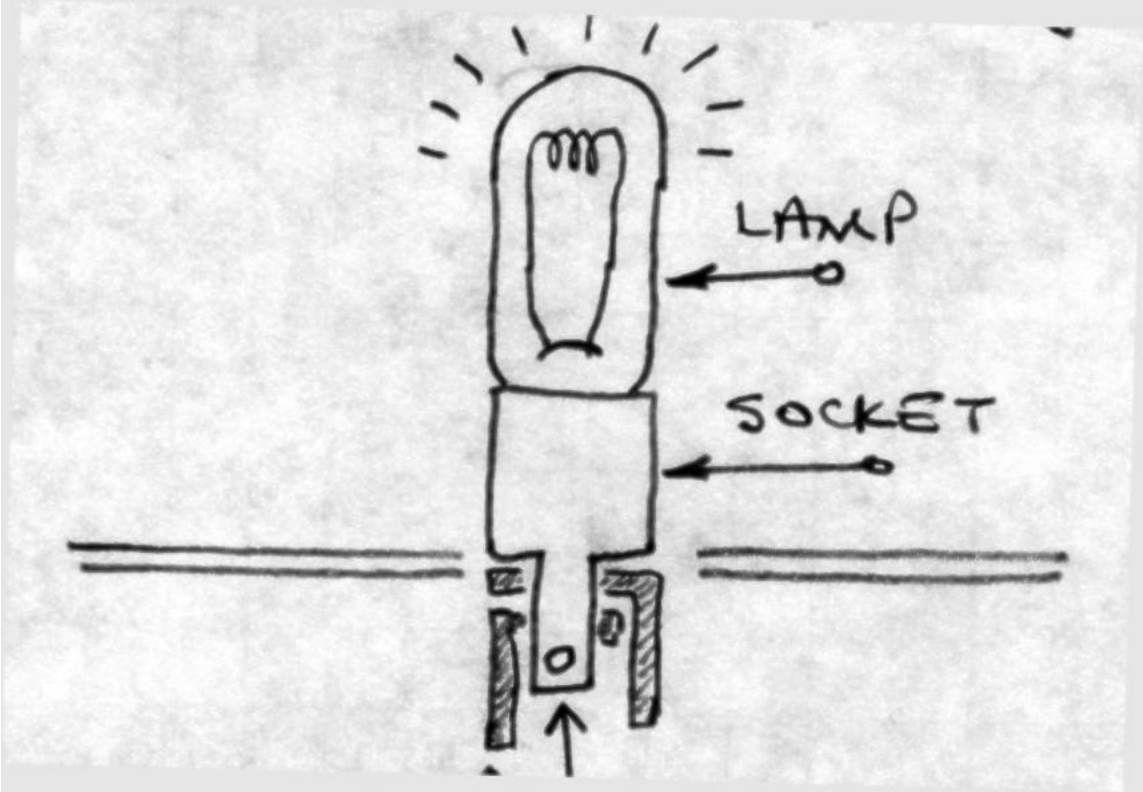
8 Electrical Power

Each relay and indicator lamp can take a significant amount of electrical current compared to today's standards for electronic circuitry. If you use relays similar to the ones I found, you'll need a 2 amp DC supply. The circuit is a bit particular about its operating voltage, so an adjustable supply is best. I found that the counter works reliably over a range of supply voltages.

9 Historical Relevance

No claim is made that this circuit is in any way historically accurate or represents circuits that were actually used in relay computers. It simply demonstrates that a sequential logic circuit can be constructed with a small number of relays.

The sequential logic circuits shown in textbooks such as references [2] and [3] predate the invention of the memory-based state machine of Wilkes. As such, and because they must deal with the physical limitations of mechanical relays, there is a strong ad-hoc flavour to the sequential machine designs. In contrast, the Wilkes state machine design is a general purpose system that can be used to generate any state machine by changing the contents of the state memory. It is interesting to speculate whether the concept of the memory-based state machine would have influenced the design of relay logic computers, had it been available.



References

[1] The Scientific American Book of Projects for the Amateur Scientist
C.L.Strong
Simon and Schuster, 1960
Pages 377-398

[2] The Design of Switching Circuits
William Keister, Alistair E. Richie, Seth H. Washburn
D. Van Nostrand, 1951

[3] Logical Design of Electrical Circuits
Rene A. Higonnet, Rene A. Grea
McGraw-Hill Book Company, 1958

Production Notes

The drawings were scanned to computer files in the `jpeg` format using a Microtek flatbed scanner attached to a Windows computer. Photographs were transferred from a Nikon Coolpix 800 digital camera into computer files in the `jpeg` format, also under the Windows operating system. The photograph and drawing files were then transferred to a computer using the Linux operating system.

On the Linux machine, photographs and drawings originally in `jpeg` format were converted to postscript format using the `xv` image manipulation program and incorporated into the text in `ps` (ecapsulated postscript) format using the `epsfig` command.

The text was edited with the `joe` editor and typeset using the LaTeX typesetting program into the `times` font.

The resultant typeset document was previewed with `xdvi`. When completely debugged, it was converted to postscript format using the program `dvips` then to `pdf` format using the script `ps2pdf`.

Fin